REMARKS/ARGUMENTS

Applicant respectfully responds to the Office Action of October 27, 2004 and requests reconsideration of the application. This paper is filed with a timely Request for Continued Examination.

Claims 1-44 are now pending, a total of 44 claims. Claims 1-30 are allowed, and claim 34 is indicated as allowable. Of the claims not allowed, claims 31 and 38 are independent.

The amendment to the specification clarifies the meanings of the terms "process" and "thread" as used herein. Such clarification of vocabulary, drawing on the knowledge in the art,[1] is not "new matter." *See Schering Corp. v. Amgen, Inc.*, 222 F.3d 1347, 1352-53, 55 USPQ2d 1650, 1654 (Fed. Cir. 2000).[2]

I. Claim 31

Claim 31 is discussed in paragraph 4 of the Action of October 2004, and paragraph 7 of the Action of March 2004, in the context of Hammond '686 alone. Claim 31 recites as follows:

> 31. A method, comprising the steps of:
>
> in response to an exception raised while executing a thread of a program coded in instructions of a first instruction set architecture, delivering the exception to an execution thread for execution of a handler for the exception, the handler's thread being distinct from the thread in which the program was executing, the handler's thread being an execution thread under an operating system coded in instructions of a second instruction set architecture, the handler being a handler of the operating system.

---

[1] See, for example, the dictionary definitions of "thread" submitted earlier in prosecution, and the discussion of "task control blocks" submitted herewith an an Exhibit.

[2] For avoidance of doubt, the definitions of the terms "thread" and "process" added by amendment are intended only to emphasize the reliance on the meanings in ordinary and actual use in the computer architecture, operating systems, and programming languages arts, to the exclusion of informal definitions that were never intended to precisely or accurately reflect the ordinary meaning actually used in the art. In any given context, the terms "process" and "thread" have extremely precise meanings, that may include requirements in addition to those stated here, or (especially in the case of technology arising near or after the filing date of this application) may be implemented in a way that presents some variation from the definitions added by amendment. Nonetheless, in any given context, those of ordinary skill define "process" and "thread" with great precision, and the definitions stated here are reasonably precise within that context.

Claim 31 recites "delivering the exception to an execution thread for execution of a handler for the exception, the handler's thread being distinct from the thread in which the program was executing."

The Office Actions compare the "execution thread" of claim 31 to elements "500a-i" of Hammond '686, and assert that a "thread" is identical (in a § 102 sense) to a "handler" in the Hammond '686 reference. Equating these two terms is incorrect, in a number of respects.

Hammond '686 discusses "handlers" 500 at col. 9, lines 52 to col. 10, line 8, and at col. 10, lines 43-65. A conventional "handler" operates as follows. When a program is interrupted, execution is transferred to the handler. Almost by definition, when a program raises an exception and invokes a handler, the part of the program that raised the exception must not be allowed to execute until the handler for that exception completes. Leaving room for rare or artificially-constructed exceptions, if an excepted program is allowed to resume execution before the handler for the exception completes, the result will be incorrect execution. Therefore, conventionally, any possibility of prematurely executing the program is removed, by executing the handler in the same thread as the program itself. Because the original context of the interrupted program is replaced with the context of the handler, the interrupted program is prevented from further execution until the handler completes. When the handler completes its job of resolving the interrupt condition, it gives up control of the thread. The scheduler eventually picks the thread for execution, and the originally interrupted program resumes as a matter of course. Thus, need and convenience nicely align in this conventional approach. Hammond '686 says nothing that would suggest that he abandoned the advantages of this conventional approach.

Claim 31 and Hammond '686 contrast in several ways. First, claim 31 juxtaposes the two terms, for example reciting "the handler's thread." If the two terms were co-extensive, as the Office Action suggests, this would make no sense. Second, under the understanding in the art, and as stated in the specification, a "thread" provides a mechanism for "independent" execution. Hammond '686 discusses no handler that executes "independently" of the program, and therefore no "thread." Third, claim 31 requires two threads to be concurrently extant – the

interrupted "program," and the initiated "thread" that handles the exception.[3] Hammond '686 never mentions the word "thread" at all, let alone more than one.

Claim 31 is patentable over Hammond '686.

II. Claims 32, 33, 35 and 37-43

Claim 38 recites language similar to claim 31, and is patentable for similar reasons.

Claims 32, 33, 35, 37, and 39-43 are dependent on the independent claims discussed above, and patentable therewith. In addition, the dependent claims recite additional features that further distinguish the art.

The amendments to the claims are not narrowing. For example, a "thread" is inherent in "executing a program." The definition of "thread" in the specification merely states the ordinary understanding in the art, and has no effect on the scope of the claims.

Nor are the amendments directed to any statutory ground of rejection. Because the issues raised in the Office Actions are non-statutory, the amendments are likewise non-statutory.

III. Inventorship

Several of the inventors are no longer easily contacted. Appropriate papers to correct inventorship will be filed when the necessary information can be gathered.

IV. Observations on Advisory Action of April 22, 2005

The reasoning of the Advisory Action of April 22 is incorrect in two respects.

First, it alleges that Applicant's "arguments focus solely upon one of the two cited dictionary definitions of thread, to the complete and absolute exclusion of the second cited dictionary." This is simply untrue. Pages 4-7 of the paper of March 30, 2005 are directed specifically to showing that the Examiner's definition of "thread" is both incorrect and irrelevant. These pages show that gluing together two informal definitions does not result in a correct definition. Three-and-a-half pages is hardly "complete and absolute exclusion."

---

[3] To clarify the record for purposes of any future litigation: as a practical matter, in most cases, the thread for the interrupted process must not be allowed to resume execution until the handler completes. Claim 31 requires only that control be transferred to a handler "thread," not necessarily that the interrupted thread and the handler thread be allowed to actually execute concurrently.

Second, the Advisory Action suggests that dictionary definitions can be relied on willy-nilly, without concern for the requirements of MPEP §§ 2111 and 2111.01. The Advisory Action suggests that *In re Morris* , 44 USPQ2d 1023 (Fed. Cir. 1997) overruled §§ 2111 and 2111.01. That legal understanding is wrong. *Morris* does not create an exception to §§ 2111 and 2111.01; indeed, § 2111 itself cites *Morris* to reinforce that claims are interpreted as the "broadest reasonable interpretation ... consistent with the interpretation those skilled in the art would reach." *Morris* itself states in the clearest terms that it intended no change to the long-established claim principles enunciated in §§ 2111 and 2111.01 (emphasis added):

> [W]e reject [the] invitation ... to overrule, *sub silentio*, decades old case law. ... As an initial matter, the PTO applies to the verbiage of the proposed claims the broadest reasonable meaning of the words in their ordinary usage as they would be understood by one of ordinary skill in the art, taking into account whatever enlightenment by way of definitions or otherwise that may be afforded by the written description contained in the applicant's specification.

44 USPQ2d at 1027. *Morris* merely holds that where a claim term has two or more correct definitions, both consistent with the specification and the understanding in the art, the PTO is to apply the "broadest reasonable interpretation" from among those correct definitions. But where a definition is simply incorrect, as here, neither *Morris* or any other precedent permits reliance on that incorrect definition. Indeed, The Board of Appeals has held that such reliance is impermissible.[4]

The Examiner makes no showing that that the definition he relies on is either correct, reasonable, or consistent with the ordinary and customary understanding of one of ordinary skill in the art – and by silence apparently concedes that his definition is none of these. The Advisory Action states no legal principle to overrule *Morris* or the Director's instructions, nor any basis to believe that the Examiner's extrapolated definition meets the appropriate legal test.

In any event, Applicant believes that any reliance on the Microsoft Computer Dictionary is mooted by the amendment to the specification.

---

[4] *E.g.*, *Ex parte Hollingsworth*, App. No. 96-2862, www.uspto.gov/web/offices/dcom/bpai/decisions/fd962862.pdf at 10-11 (BPAI 1996) (examiner's reliance on dictionary is "indiscriminate," "absurd," and "beyond all reason;" Board reminds that a claim must be interpreted "in a manner consistent with the specification and construed as those skilled in the art would construe them")

If the Examiner believes himself exempt from the Director's instructions in MPEP § 2111.01, Applicant requests a written statement from some Patent Office official who has authority to create such an exemption. Without such a written statement, Applicant respectfully requests that the Examiner observe those instructions.

If the Examiner believes that his interpretation of "thread" is "reasonable" and consistent with the understanding of those of ordinary skill, Applicant requests a reference showing actual usage of the word "thread" to include "everything from a single instruction, to a conventional subroutine, to the purely hardware 'process' of changing a transistor state from 0 to 1, to loading the operating system into memory on cold boot before the operating system's thread mechanisms are even initialized, to any mechanism for handling an interrupt" as discussed at Applicant's paper of March 30, 2005, at page 7.

V. Conclusion

In view of the amendments and remarks, Applicant respectfully submits that the claims are in condition for allowance, and requests that the application be passed to issue in due course. The Examiner is urged to telephone Applicant's undersigned counsel at the number noted below if it will advance the prosecution of this application, or with any suggestion to resolve any condition that would impede allowance. Kindly charge any additional fee, or credit any surplus, to Deposit Account No. 23-2405, Order No. 114596-28-000053BS.

Respectfully submitted,

WILLKIE FARR & GALLAGHER LLP

Dated: April 27, 2005

By: _____

David E. Boundy
Registration No. 36,461

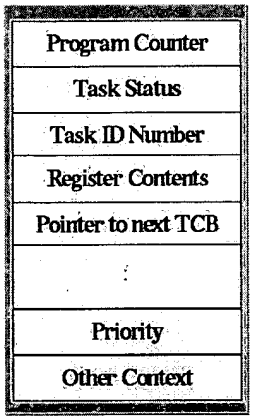
WILLKIE FARR & GALLAGHER LLP
787 Seventh Ave.
New York, New York 10019
(212) 728-8757
(212) 728-9757 Fax

2.3 Task Dispatching

Task Control Blocks: A common method of mananging the "bookkeeping" associated with task creation, scheduling and synchronization in Real-time Operating Systems is the Task Control Block. This is a structure in memory that holds all the pertinent information for a task. The figure below shows a typical Task Control Block (TCB).

- Task-control blocks is the most popular method of identifying and managing tasks.
- Task-Control Block (TCB) contains:
  - a context (e.g. program counter and register contents)
  - an identification string
  - a status, such as ready, executing or blocked
  - a priority (if applicable)

A Typical Task Control Block



Task States: The operating system manages the Task Control Blocks by keeping track of the status or state of each task.

A task can typically be in any one of the following states:

- Executing
- Ready
- Suspended

Executing state - the task that is actually running
Ready state - tasks that are ready to execute
Suspended (of blocked) state - tasks that are waiting on a particular resource, and thus are not ready

---

Table of Contents